Introduzione alle reti neurali per il riconoscimento di immagini

Una sintesi delle tecniche sviluppate

Alessandro Falaschi

Dipartimento di Ingegneria dell'Informazione, Elettronica e Telecomunicazioni - DIET
Università La Sapienza di Roma

Pubblicato su TeoriadeiSegnali.it

Contributo al corso di Advanced Electromagnetics and Scattering

Prof. Fabrizio Frezza

Maggio 2021

Introduzione

La presentazione ha luogo nel contesto del corso
Advanced Electromagnetics and Scattering
svolto al secondo anno della laurea magistrale in *Ingegneria Elettronica* ed al primo di quella in *Atmospheric Science and Technology* presso rispettivamente l'Università Sapienza di Roma e quella dell'Aquila

L'intento è quello di fornire una introduzione alle basi teoriche delle reti neurali, con particolare riguardo per quelle convoluzionali, facendo uso di materiale didattico disponibile in rete come di strumenti di visualizzazione interattiva.

La presentazione ha dunque il solo scopo di fare da collante per coordinare i rimandi al materiale utilizzato

Indice dei contenuti

Ora parliamo di...

1. **Reti neurali convoluzionali**
   - Dalla biologia alla struttura di calcolo
   - Multi Layer Perceptron
   - Discesa del gradiente e back-propagation
   - Deep learning
   - Reti convoluzionali
   - Il resto del mondo

Il neurone biologico

- Il cervello umano contiene quasi 100 miliardi di neuroni ciascuno dei quali connesso con 500 - 1000 altri neuroni
  - totale $\simeq 10^{14}$ collegamenti sinaptici!
  - il numero dei neuroni resta sostanzialmente quello della nascita, mentre il numero delle sinapsi può variare per tutta la vita
- i segnali in ingresso ad un neurone sono *treni di impulsi* (elettrici), e quando la frequenza di arrivo eccede una soglia, il neurone produce segnali della stessa natura
- la *corteccia cerebrale* (2 - 5 mm) ospita 16 miliardi di neuroni disposti su sei strati, ed è una esclusiva dei mammiferi
- le reti neurali *artificiali* emulano le sinapsi ed i treni di impulsi mediante una combinazione lineare di valori continui

I riferimenti principali

Anziché ripetere cose già scritte da altri, adotto il seguente materiale, a cui riferirsi via via

Michele Scarpiniti - rif.A

Neural Networks presso questo link

pagg. 4 - 9

Davide Maltoni - rif.B

Reti Neurali presso questo link

pagg. 2 - 7

Algebra lineare? Guardiamola!

- Qualunque corso di Machine Learning include un riepilogo dei risultati di algebra lineare e statistica
  - In questo che è un seminario e non un corso, ci limitiamo a visualizzare i risultati rilevanti con l'aiuto di geogebra
- la combinazione lineare (o *prodotto scalare*)

$$s = \boldsymbol{w}^T \boldsymbol{x} = \sum_{i=0}^{M} w_i x_i$$

degli ingressi x_i (più *il bias* $x_0 = 1$) individua l'equazione di un (iper)piano che suddivide il piano (lo spazio) descritto dagli $x_{1:M}$ in due regioni. I possibili valori di $\boldsymbol{x}$ danno un valore $s \gtrless 0$ a seconda se il *punto* corrispondente sia *sopra o sotto* la retta - *Rif.* ***A*** *pp. 10 - 11*

- la scelta dei *pesi* w che definiscono l'equazione che separa due classi si basa sulla *discesa del gradiente* (la affrontiamo a breve) che inizializza w_i con valori casuali ed iterativamente li modifica per minimizzare l'errore quadratico medio $e^2 = \overline{(d - s)^2}$ in cui $d \gtrless 0$ è il valore desiderato per s in corrispondenza di un ingresso $\boldsymbol{x}$

Ora parliamo di...

1. **Reti neurali convoluzionali**
   - Dalla biologia alla struttura di calcolo
   - Multi Layer Perceptron
   - Discesa del gradiente e back-propagation
   - Deep learning
   - Reti convoluzionali
   - Il resto del mondo

Per ogni neurone, un piano nello spazio

- Se il dataset riguarda più di due classi occorre più di un piano e quindi di un neurone
  - sempre con l'aiuto di geogebra verifichiamo che possiamo individuare solamente regioni *linearmente separabili*
  - il che impedisce di gestire dataset più *contorti*
- la soluzione passa per due aggiunte
  - l'utilizzo di *più di un livello* di neuroni (*Multi Layer Perceptron* o MLP)
  - una *non linearità* $\varphi(s)$ o *funzione di attivazione* (*AF*) applicata per ogni neurone al risultato della somma $s = \boldsymbol{w}^T \boldsymbol{x}$ - *Rif.* ***A*** *pp. 14 - 19*
    - ★ mima gli effetti di saturazione e non linearità dei neuroni naturali
    - ★ migliora la precisione di calcolo mantenendo i valori limitati
    - ★ permette la dimostrazione di un teorema che afferma la possibilità di approssimare qualunque funzione
    - ★ altrimenti non importa quanti sono i livelli, senza AF sarebbe come avere un livello solo
- *Rif.* ***A*** *pp. 20 - 29*

Quei pesi migliori del mondo

Nel senso di Ottimi

- I *pesi* $\boldsymbol{w}$ sono scelti in modo che l'errore quadratico $\overline{e^2(\boldsymbol{w}, \boldsymbol{x})} = \overline{\left|f(\boldsymbol{x}) - \hat{f}(\boldsymbol{w}, \boldsymbol{x})\right|^2}$ tra il valore desiderato $f(\boldsymbol{x})$ e quello $\hat{f}(\boldsymbol{w}, \boldsymbol{x})$ in uscita dal MLP *sia minimo*
  - viene adottato un metodo iterativo noto come *discesa del gradiente*
- il gradiente ∇f di una funzione f di più variabili è un vettore che estende il concetto di derivata nello spazio $n-$dimensionale (Geogebra)
  - la sua $i-$esima componente è la derivata parziale della funzione rispetto alla variabile x_i
  - è orientato nella direzione verso la quale la funzione *cresce di più*
    - ★ quindi per andare verso *il minimo* basta muoversi in direzione *opposta* (Geogebra)
  - ci si *accorge* di essere arrivati al minimo perché lì il gradiente *si annulla*
    - ★ a meno di non incappare in un minimo *locale*, od in punto *di sella*

Ora parliamo di...

1. **Reti neurali convoluzionali**
   - Dalla biologia alla struttura di calcolo
   - Multi Layer Perceptron
   - Discesa del gradiente e back-propagation
   - Deep learning
   - Reti convoluzionali
   - Il resto del mondo

La discesa del gradiente

e la Back Propagation dell'errore

- Attingiamo ad una ulteriore risorsa

Michele Scarpiniti - rif.C

Introduction to optimization presso questo link -> pagg. 8 - 11

- Essendo l'uscita del MLP una funzione di funzioni di funzioni (etc...) per trovare le derivate dell'errore e^2 rispetto ai pesi $\boldsymbol{w}$ dei livelli precedenti all'ultimo occorre applicare le regole della derivazione *a catena*

$$\frac{d}{dx}[f(g(x))] = f'(g(x)) \cdot g'(x)$$

  su cui si basa l'algoritmo di *retropropagazione dell'errore*, illustrato presso
  - *Rif.* ***A*** - pagg. 12 - 13, 21 e 30 - 35
  - *Rif.* ***B*** - pagg. 15 - 19
- è stata una *fortuna* aver scelto le $\varphi(s)$ derivabili!

Modifica dei pesi

- La discesa del gradiente *stocastico* fu introdotta nel '60 come metodo di sintesi per i coefficienti di un filtro FIR di cui si conosce l'uscita desiderata, per realizzare ad esempio un cancellatore d'eco od un equalizzatore adattivo - riferimento teoriadeisegnali.it pag. 27
- per ogni ingresso $\boldsymbol{x}^0[n]$ ogni peso w_{ij}^l tra il neurone i al livello $l-1$ ed il neurone j al livello l viene aggiornato come

$$w_{ij}^l[n] = w_{ij}^l[n-1] + \mu \cdot \delta_j^l[n] \cdot x_i^{l-1}[n]$$

in cui $\delta_j^l[n] = -e_j[n]\,\varphi_j'(s_j[n])$ è il prodotto (con il segno cambiato) tra l'errore e_j sul neurone j di destinazione e la φ_j' calcolata sui valori *in ingresso* al neurone

  - se l è il livello di uscita, l'errore $e(j)$ si ottiene per via diretta
  - altrimenti $e(j)$ dipende dagli errori ai neuroni *dello strato successivo* $l+1$, moltiplicati per *i pesi* relativi

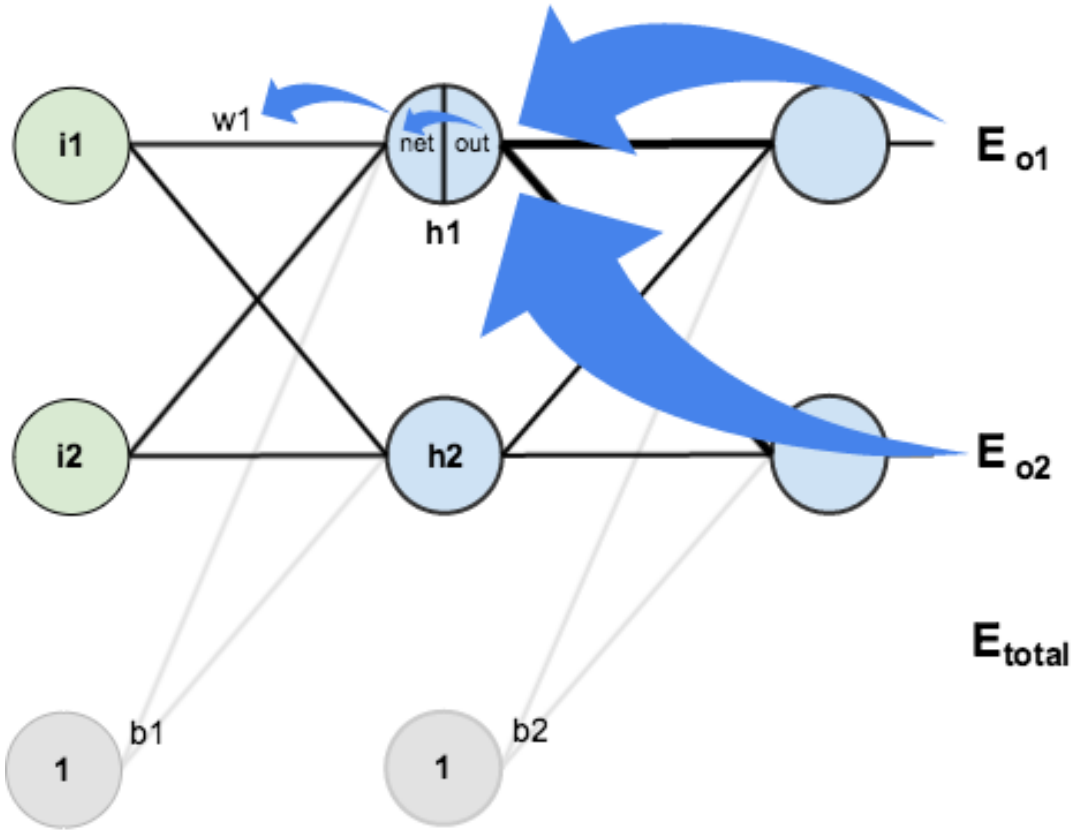
All'interno dei livelli nascosti

Tratto da A Step by Step Backpropagation Example

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\downarrow$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



Andiamo in scena

- una rete neurale *allenata* su di un dataset $(\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1,\cdots,N}$ funziona come
  - un *regressore* $\hat{\boldsymbol{y}} = \hat{f}(\boldsymbol{w}, \boldsymbol{x})$ per nuovi ingressi $\boldsymbol{x}$
    - ★ con i neuroni di uscita in numero pari alla dimensione di $\boldsymbol{y}$ e senza non linearità, in modo da minimizzare $\overline{\left(\boldsymbol{y}_i - \hat{f}(\boldsymbol{w}, \boldsymbol{x}_i)\right)^2}$
  - un *classificatore* quando y_i individua *una categoria* per x_i mediante codifica one-hot, con i neuroni di uscita pari al numero delle classi
    - ★ con uno strato *softmax* per tradurne le attivazioni in *probabilità*
    - ★ con una funzione costo di entropia mutua tra le $\hat{p}_j$ delle classi e quelle della codifica one-hot di $\boldsymbol{y}$
    - ★ nel caso di due sole classi, in uscita basta un neurone con $\varphi(s)$ sigmoide
- un bellissimo simulatore interattivo in javascript si trova presso
  - http://playground.tensorflow.org
  - permette di seguire il processo di apprendimento potendo variare numero di livelli, di neuroni, funzioni di attivazione, vincoli di regolarità...

Ora parliamo di...

1. **Reti neurali convoluzionali**
   - Dalla biologia alla struttura di calcolo
   - Multi Layer Perceptron
   - Discesa del gradiente e back-propagation
   - Deep learning
   - Reti convoluzionali
   - Il resto del mondo

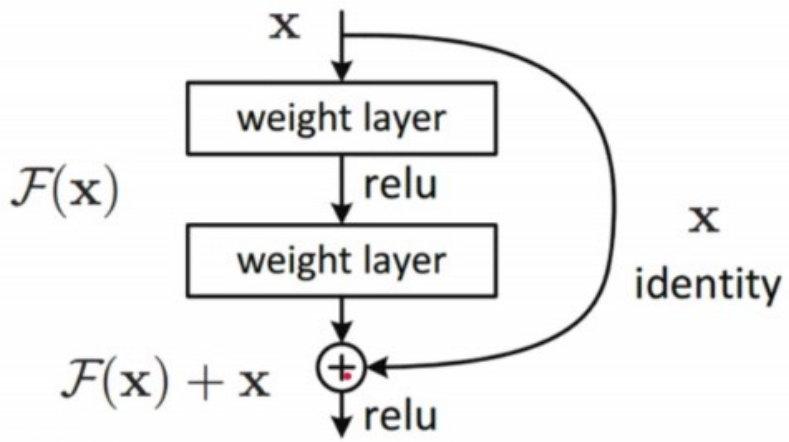
E' qui che entrano in gioco i big data

- da un punto di vista statistico, all'aumentare del numero di parametri da stimare aumenta la quantità di dati necessaria ad ottenere stime adeguate
- con il progredire degli accorgimenti tecnici e della capacità di calcolo si è divenuti in grado di analizzare le quantità di dati ormai presenti in rete
- *Rif.* ***A*** *pp. 43 - 54*

Trucchi ed espedienti

A seguito dell'apprendimento si può verificare **overfitting** quando i parametri divengono *troppo legati* ai dati di apprendimento, e viene a mancare la capacità di *generalizzare*. Possibili rimedi sono

- *dropout* e normalizzazione
  - Rif. A pp. 64 - 67
- regolarizzazione
  - nella forma più basilare, l'aggiunta alla funzione obbiettivo e di un termine (*regolarizzatore*) proporzionale alla somma dei pesi, in termini assoluti (norma L_1) o quadratici (norma L_2)
- residual learning
  - si aggiungono connessioni che *scavalcano* un livello e *si sommano* alla sua uscita, permettendo ai neuroni di apprendere $f(\boldsymbol{x}) = h(\boldsymbol{x}) - \boldsymbol{x}$ anziché $h(\boldsymbol{x})$



L'insieme di verifica

Probabilmente il miglior modo per evitare l'overfitting:

- gli esempi nel *dataset* D a disposizione sono ripartiti in tre sottoinsiemi disgiunti
  - l'insieme di *training* D_{tr}
  - quello di *validazione* D_v
  - e quello di *test* D_{te}
- le iterazioni (passo forward + passo backward) del processo di apprendimento prendono il nome di *epoche* e fanno uso di tutti gli esempi di D_{tr}
  - le epoche sono suddivise in *mini-batch*, ognuno dei quali utilizza un diverso sottoinsieme di D_{tr} su cui *accumulare* i valori della funzione costo, ed al termine dei quali avviene *l'aggiornamento* dei pesi
- al termine di ogni epoca il modello è usato per classificare gli esempi degli insiemi D_{tr} e D_v
  - mentre D_{tr} continua a diminuire, D_v può invece aumentare, segnalando l'overfitting

Ora parliamo di...

1. **Reti neurali convoluzionali**
   - Dalla biologia alla struttura di calcolo
   - Multi Layer Perceptron
   - Discesa del gradiente e back-propagation
   - Deep learning
   - Reti convoluzionali
   - Il resto del mondo

Le ragioni di un nome

- sicuramente abbiamo già sentito parlare di *convoluzione*: è *il funzionamento* di un filtro (o sistema lineare e permanente) per produrre in uscita un valore y che dipende
  - dalla memoria degli ingressi x fino a quel momento
  - dalla risposta impulsiva $h(t)$ del filtro
- nel caso di sequenze la convoluzione FIR $y_n = \sum_{k=n}^{n-N} x_k h_{n-k}$ esegue la stessa operazione (un *prodotto scalare*) che avviene all'ingresso di ogni neurone, con i valori $h_{n,k}$ (il *kernel*) al posto dei $w_{n,k}$
- esiste poi il filtro *adattato*, con una h che si ottiene *ribaltando* il segnale x che si intende *rivelare*, e che permette di distinguerlo anche con molto rumore - rif. teoriadeisegnali.it p. 24
- i pesi w collegati ad un neurone sono interpretabili come una h adattata a rivelare... *segnali scoperti durante il processo di apprendimento!*
- nel caso di segnali bidimensionali (le immagini) anche le h sono 2D, così come i neuroni di un livello sono disposti a matrice, la *feature map*, una per ogni filtro

Reti convoluzionali

dalla teoria alla pratica

- *Rif.* ***A*** *pp. 68 - 73*
- animazione interattiva di un classificatore di immagini in javascript
- visualizzatore 3D della CNN per il riconoscimento di cifre

Ora parliamo di...

1. **Reti neurali convoluzionali**
   - Dalla biologia alla struttura di calcolo
   - Multi Layer Perceptron
   - Discesa del gradiente e back-propagation
   - Deep learning
   - Reti convoluzionali
   - Il resto del mondo

Non abbiamo parlato di...

- **Autoencoder** - architetture multilivello non supervisionate, allenate a riprodurre gli ingressi dopo averne ridotto la dimensionalità. Adatte per il *denoising* e come *pre-training* di strati di architetture profonde
- **Reti ricorsive** - adatte per l'analisi di sequenze, le uscite dipendono oltre che dagli ingressi, anche dallo stato interno della NN
- **Reti generative *avversarie*** - prevedono due reti, di cui
  - una *genera* immagini sintetiche a partire da rumore gaussiano
  - l'altra *discrimina* sia immagini reali che sintetiche, come vere o false
  - il discriminatore è addestrato per la max prob. di decisione corretta, mentre il generatore per la max prob. di ingannare il discriminatore
  - il risultato sono immagini sintetiche realistiche
- **Transformers** - nati nel contesto della *traduzione automatica*, si applicano a riconoscimento e sintesi vocale, e vari altri casi
  - possono adottare stadi di elaborazione convoluzionale e ricorsiva
  - contemplano aspetti degli autoencoder, ma con segnali di uscita diversi da quelli di ingresso
  - la *rappresentazione interna* viene *filtrata* da un meccanismo definito *attenzione* (= ciò che *attiene*)